CURSO PRÁTICO 5 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 1 N.º 5

NESTE NÚMERO

APLICAÇÕES

ORGANIZE AS SUAS COLEÇÕES (2)

Como pesquisar, modificar, apagar e imprimir um registro..... 81

PROGRAMAÇÃO BASIC

PROGRAME JOGOS A CORES

Como acrescentar cores aos desenhos em computadores compatíveis com o TRS-Color. Aprenda a explorar os comandos **PMODE**..... 86

CÓDIGO DE MÁQUINA

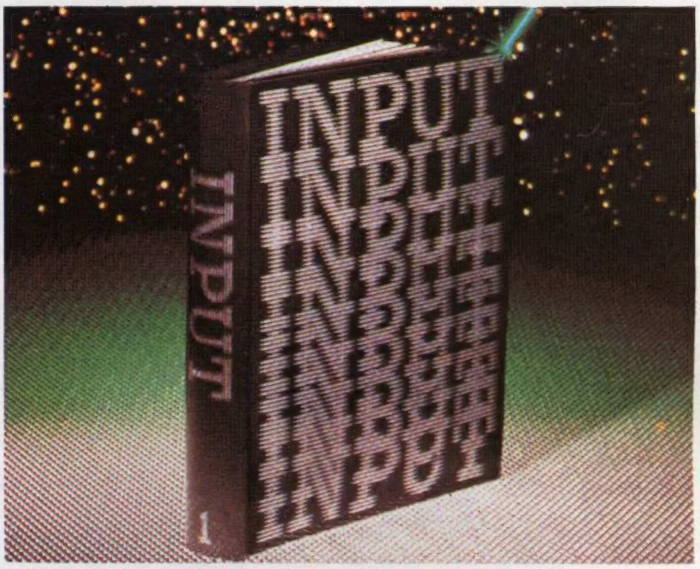
COMO ENTRAR CÓDIGO DE MÁQUINA

Trabalhe com código de máquina, usando um programa em BASIC e a instrução **POKE**. O que são e para que servem os monitores..... 88

PROGRAMAÇÃO BASIC

O QUE SÃO VARIÁVEIS

Como trabalhar com variáveis. O que é um cordão. Como os cordões vazios são empregados na programação de jogos..... 96



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o n.º (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.
Tradução: Aluísio J. Dornellas de Barros, Maria Fernanda Sabbatini
Adaptação, programação e redação: Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.
Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

ORGANIZE AS SUAS COLEÇÕES (2)

- COMO PESQUISAR E MODIFICAR UM REGISTRO
- IMPRIMA O ARQUIVO
- UTILIZE O PROGRAMA PARA SEUS ARQUIVOS

Encontrar o nome e o endereço de uma pessoa da qual se sabe apenas o sobrenome e a cidade em que reside é tarefa simples para o computador. Mas, para isso, é preciso programá-lo com as informações adequadas.

Uma das grandes vantagens do arquivamento no computador, em comparação com métodos mais tradicionais (como um fichário, por exemplo), é a facilidade e rapidez com que se pode encontrar as informações arquivadas.

O programa da lição anterior da série *Aplicações* fornecia os meios para se implementar um sistema altamente flexível de arquivamento no computador. Neste artigo mostramos como se pode pesquisar arquivos, tendo em vista objetivos como a correção de registros desatualizados ou incorretos, ou a anulação de registros desnecessários.

COMO PESQUISAR UM ARQUIVO

A opção de pesquisa é a de número 4 no menu principal. Ela permite que você pesquise os registros a partir da informação contida em um campo.

Se você pressionar a tecla 4, o computador mostrará, no alto da tela, os nomes dos campos que você definiu anteriormente para o arquivo; em seguida, ele perguntará qual campo você quer pesquisar — 1 para o primeiro, 2 para o segundo, 3 para o terceiro e assim por diante, contando a partir do alto da tela. A pergunta seguinte será sobre o que você quer pesquisar nesse campo. Em resposta, você deve digitar a palavra ou o número que quer procurar: JOÃO ou CAÇAROLA, por exemplo, ou BÍBLIA, ou seja lá o que for. Em seguida pressione <**RETURN**> ou <**ENTER**>.

A palavra ou o número que você digitar deve ser exatamente igual ao que está escrito no campo. Se uma palavra foi armazenada com letras maiúsculas nos registros e você estiver procurando-a com letras minúsculas, o computador não irá encontrá-la. Se você tiver deixa-

do, por engano, um espaço antes da entrada em um registro, ou apertar o espacejador após a entrada, provavelmente o computador não irá encontrá-la. Isso ocorre mesmo que a palavra do registro e a que você está procurando sejam iguais.

Assim, evite espaços desnecessários e procure entrar informações em letras maiúsculas. A pesquisa se tornará mais fácil e menos sujeita a erros.

Caso o computador não consiga achar nenhum registro com a palavra que você pediu, no campo pesquisado, ele lhe dirá isso e trará de volta o menu principal. Se, ao contrário, forem encontradas diversas respostas, ele colocará na tela o primeiro registro que tiver a informação pedida e, a partir daí, listará em ordem alfabética todos aqueles com informação igual.

Você deve então selecionar uma das opções colocadas nas duas linhas abaixo da tela. A primeira é assim:

```
PROSSEGUE    RETORNA    MENU
```

Isso funciona mais ou menos da mesma maneira que no programa do artigo anterior. A tecla P serve para pesquisar os registros selecionados em ordem crescente, ao passo que a tecla M o leva diretamente de volta ao menu principal. Mas não acontecerá nada se você pressionar a tecla P quando estiver no último registro selecionado. A tecla R, na primeira modalidade de pesquisa, fornece uma mensagem: "REGISTRO NÃO ENCONTRADO".

Talvez a aplicação mais útil da opção de pesquisa seja descobrir um determinado registro. Para localizá-lo, basta lembrar-se de qualquer informação sobre ele. Por exemplo, suponhamos que você esteja utilizando esse arquivo para armazenar nomes, endereços e números de telefones dos membros de seu clube e já possua os seguintes campos: nome, sobrenome, rua, cidade, Estado,código postal, número do telefone. Neste caso, você poderá pesquisar um determinado registro até mesmo se lembrar apenas do sobrenome da pessoa.

E se uma parte da informação de que você se lembrou não for exclusiva desse registro? A solução é simples: por exemplo, se você procura o registro de uma pessoa, sabendo apenas que ela mora na cidade de São Paulo, ou que o seu primeiro nome é João, poderá evocar todos os paulistanos ou todos os indivíduos de nome João e saltá-los até encontrar o nome correto. Mesmo que o número de nomes fosse muito grande, essa procura seria mais fácil do que se tivesse que percorrer todo o arquivo, revisando um registro de cada vez.

COMO MODIFICAR UM REGISTRO

A segunda linha de opções da parte inferior da tela permite outras opções de trabalho com o registro exibido:

```
CORRIGE    APAGA    IMPRIME
```

Se você quiser modificar um registro mostrado na tela, deve pressionar a tecla C. O computador perguntará "QUAL O NÚMERO DO CAMPO" que você deseja modificar. Digite o número, contando a partir do alto.

Em seguida, o computador pedirá para "ENTRAR A MODIFICAÇÃO". Você deve então digitar o novo campo que quer entrar, até mesmo se quiser modificar apenas uma letra. Quando as teclas **<ENTER>** ou **<RETURN>** forem pressionadas, o computador efetuará a modificação no lugar correto do registro. Se você quiser, poderá pressionar mais uma vez a tecla C e modificar ou-

tro campo no mesmo registro. Se você quiser modificar um registro que não está na tela, deve pressionar C para retornar ao menu principal e escolher a opção de pesquisa 3, para localizar o registro que deseja alterar. Outra saída seria escolher a opção 2, "VER OS REGISTROS", achando-o e modificando-o.

APAGUE OS REGISTROS INCORRETOS

Você deve primeiro localizar o registro que deseja apagar, estudando as alternativas. Normalmente, são apresentadas, no lado inferior da tela, seis opções padrão. Para apagar um registro é necessário pressionar a tecla A. Então o computador perguntará: "VOCÊ TEM CERTEZA?". Isso evita que você apague, por engano, um registro còrreto.

Se você tem certeza de que quer apagá-lo, pressione a tecla S. O computador apagará então esse registro e exibirá o próximo — tanto o registro anotado segundo uma ordem alfabética, se você estiver na modalidade de exibição, quanto o que possui o mesmo campo que você está pesquisando, se estiver na modalidade de pesquisa.

Se você apagou o último registro encontrado na pesquisa, o computador informará que não existem mais registros com tal item no campo que o interessa e trará o menu principal de volta.

IMPRIMA OS REGISTROS

A última opção na parte inferior da tela permite mandar para a impressora (caso haja uma) os registros do arquivo. Se você pressionar a tecla I na versão do programa para o TRS-Color, o computador lhe pedirá para "CHECAR A IMPRESSORA". Nesse ponto, é necessário averiguar se a impressora está conectada e ligada; se essa condição for satisfeita e mesmo assim a impressora não funcionar, você deverá verificar o programa.

No Spectrum essa linha de mensagem não é necessária, pois ele não reagirá à tecla I se não houver nenhuma impressora conectada.

Uma vez que você já checou se a impressora está conectada e ligada, acione a tecla C para continuar. Pressionando algumas outras letras, a linha "CHEQUE A IMPRESSORA" desaparecerá e você poderá novamente utilizar uma das seis opções padrão, localizadas no lado inferior da tela. Caso C seja pressionada sem que haja uma impressora conectada, ou esta não funcione por qualquer razão, você deve pressionar as teclas **<RESET>** (no TRS-Color e Apple), ou **<CTRL> <STOP>** (no MSX), e verificar o programa. Para continuar a utilizar o programa sem perder os dados armazenados, digite de modo direto:

```
GOTO 30
```

Retornando ao menu principal (uma observação importantíssima: não tente modificar qualquer linha do programa antes de dar o comando acima, caso contrário perderá todos os dados já armazenados na memória RAM da máquina).

SAIBA UTILIZAR SEUS ARQUIVOS

Agora você possui um sistema de arquivamento altamente flexível e completo, que pode ser utilizado para armazenar detalhadamente qualquer tipo de informação. Evidentemente, você pode utilizar o programa para armazenar, em uma única fita cassete, um fichário enorme ou vários fichários.

Fitas muito longas, contudo, apresentam inconvenientes. Uma solução mais prática consiste em armazenar diferentes arquivos em várias fitas, de modo a torná-los mais facilmente localizáveis. Neste caso, é necessário guardar o próprio programa em uma fita separada (somente os micros da linha Spectrum estão dispensados dessa precaução). Você poderá, assim, carregar os dados antes de selecionar a opção 6.

À primeira vista, isso pode parecer um grande desperdício de número de acessos. Essa impressão, contudo, deixa de ter sentido quando pensamos no espaço ocupado pelos arquivos dos antigos cartões de indexação.

COMO ENTRAR O PROGRAMA

A seguir, listamos a segunda seção do programa de arquivamento de dados que, para sua felicidade, é muito mais curta do que a primeira.

Quando acrescentá-la ao programa existente, você notará que poucos números de linhas estão duplicados. Não esquente a cabeça com isso. Os números da primeira seção estavam ali simplesmente para manter o programa intacto e permitir o seu teste. Os novos números substituirão os antigos assim que você digitá-los.

Atenção se você for utilizar o programa abaixo em micros com disquete. Mude o número 11000 do comando **CLEAR**, no começo do programa, para 8000, sob pena de não haver espaço suficiente na memória.

```
3000.GOSUB 8500
3010 PRINT @417,"MODIFICAR QUAL
 CAMPO?(1 A";A;") ?";
3020 IN$=INKEY$:IF IN$="" THEN
3020
3030 J=VAL(IN$)
3040 IF J<1 OR J>A THEN 3020
3050 PRINT @448,""
3060 PRINT @45+32*J,"":PRINT @4
17,"DIGITE O CAMPO MODIFICADO "
:LINE INPUT A$(D,J)
3070 A$(D,J)=LEFT$(A$(D,J),A(J)
)
3080 IF D=R THEN J=-1:GOTO 3130
3090 IF D=1 THEN J=1:GOTO 3120
3100 IF A$(D,1)>A$(D+1,1) THEN
J=1
3110 IF A$(D,1)<A$(D-1,1)THEN J
=-1
3120 IF A$(D+1,1)="" AND J=1 TH
EN 3180
3130 IF J=1 THEN 3160
3140 IF A$(D,1)>=A$(D-1,1) THEN
 3180
3150 FOR N=1 TO A:X$=A$(D,N):A$
(D,N)=A$(D-1,N):A$(D-1,N)=X$:NE
XT:D=D-1:GOTO 3080
3160 IF A$(D,1)<=A$(D+1,1) THEN
 3180
3170 FOR N=1 TO A:X$=A$(D,N):A$
(D,N)=A$(D+1,N):A$(D+1,N)=X$:NE
XT:D=D+1:GOTO 3080
3180 FOR F=1 TO 300:NEXT:RETURN
4000 G=D
4010 GOSUB 8500
4020 PRINT @449,"VOCE TEM CERTE
ZA QUE QUER APAGAR?"
4030 IN$=INKEY$:IF IN$="" THEN
4030
4040 IF IN$<>"S" THEN RETURN
4050 IF G=NR THEN G=G-1
4060 CLS 7:PRINT @236,"APAGANDO
";:SOUND 30,1
4070 IF D=NR THEN 4090
4080 FOR N=1 TO A:A$(D,N)=A$(D+
1,N):NEXT:D=D+1:GOTO 4070
4090 FOR N=1 TO A:A$(D,N)="":NE
XT:NR=NR-1:D=G
4100 FOR F=1 TO 400:NEXT:RETURN
5000 FOR N=1 TO A:PRINT @33+32*
N,N,N$(N):NEXT
5010 PRINT @417,"QUE CAMPO PROC
URAR (1 A";A;") ?";
5020 IN$=INKEY$:IF IN$="" THEN
5020
5030 IF VAL(IN$)<1 OR VAL(IN$)>
A THEN 5020
5040 SOUND 30,1:Z=VAL(IN$):PRIN
T @417,"PROCURAR PELO QUE (NO C
AMPO";Z;")?"
5050 LINE INPUT Z$
5060 D=1:G=0:CH=1
5070 IF D>NR AND G=1 THEN G=0:C
H=-1 ELSE IF D>NR THEN 5230
5080 IF D<1 AND G=1 THEN G=0:CH
=1 ELSE IF D<1 THEN 5230
5090 IF A$(D,Z)<>Z$ THEN D=D+CH
:GOTO 5070
5100 LD=D:G=1:GOSUB 8500
5110 PRINT @451,"pROSSEGUE
```

```
ETORNA   mENU    cORRIGE    aPA
GA     iMPRIME";
5120 IN$=INKEY$:IF IN$="" THEN
5120
5130 IN=INSTR(1,RS$,IN$)
5140 ON IN GOTO 5160,5160,5170,
5180,5190,5200,5210
5150 GOTO 5120
5160 CH=1:D=D+1:GOTO 5070
5170 CH=-1:D=D-1:GOTO 5070
5180 RETURN
5190 GOSUB 3000:GOTO 5090
5200 GOSUB 4000:GOTO 5090
5210 GOSUB 10000:GOTO 5070
5220 GOTO 5110
5230 CLS 2:PRINT @200,"NENHUM R
EGISTRO COM ";:PRINT @271-LEN(Z
$)/2," ";Z$;" ";
5240 PRINT @330," NO CAMPO";Z;
5250 FOR G=1 TO 5:SCREEN 0,1:FO
R F=1 TO 500:NEXT:SCREEN 0,0:FO
R F=1 TO 500:NEXT F,G:RETURN
```

```
4000 FOR N=V TO A: PRINT  INVER
SE V;AT N*2,9;N; INVERSE U;TAB
11;":  ";N$(N): NEXT N
4010 PRINT ''"PROCURAR EM QUE C
AMPO (1 A ";A;")?"
4020 IF INKEY$="" THEN  GOTO 40
20
4030 LET Y$=INKEY$: IF CODE Y$<
49 OR CODE Y$>48+A THEN  GOTO 4
030
4040 SOUND .1,10: LET Z=VAL Y$:
 PRINT '"PROCURAR PELO QUE NO C
AMPO ";Z;"?": DIM Z$(V,A(Z)): I
NPUT  LINE Z$(V)
4044 CLS : LET K=V
4045 IF A$(K,B(Z)+V TO B(Z+V))=
Z$(V) THEN  GOTO 4050
4046 IF K=R OR A$(K,V)=" " THEN
  CLS : PRINT AT 9,3;"NENHUM RE
GISTRO COM ";Z$(V),'TAB 10;"NO
CAMPO ";Z: PAUSE 150: RETURN
4047 LET K=K+V: GOTO 4045
4050 LET D=V: LET PM=V: LET MO=
V
4060 IF D>R THEN  LET D=PM
4070 IF D=U THEN  LET D=PM
4080 IF A$(D,V)=" " THEN  LET D
=PM
4090 IF A$(D,B(Z)+V TO B(Z+V))<
>Z$(V) THEN  LET D=D+MO: GOTO 4
060
4100 GOSUB 9500
4110 LET PM=D
4120 IF OP=V THEN  LET MO=V: LE
T D=D+MO: GOTO 4060
4130 IF OP=2 THEN  LET MO=-V: L
ET D=D+MO: GOTO 4060
4140 IF OP=3 THEN  RETURN
4150 IF OP=4 THEN  GOSUB 8000
4160 IF OP=5 THEN  LET DF=U: LE
T MD=2: GOSUB 9000: IF DF=V OR
A$(V,V)=" " THEN  RETURN
4170 GOTO 4100
8000 INPUT AT U,U;"CAMPO A SER
CORRIGIDO(1 A ";(A);")?";J: IF
J>A THEN  GOTO 8000
8010 PRINT  FLASH V;AT V+2*J,12
;A$(D,B(J)+V TO B(J+V)): INPUT
AT U,U;"Digite o campo modifica
do", LINE A$(D,B(J)+V TO B(J+V)
): PRINT AT V+2*J,12;A$(D,B(J)+
V TO B(J+V))
8020 IF D=R THEN  LET J=-V: GOT
O 8070
8030 IF D=V THEN  LET J=V: GOTO
 8060
8040 IF A$(D)>A$(D+V) THEN  LET
 J=V
8050 IF A$(D)<A$(D-V) THEN  LET
 J=-V
8060 IF A$(D+V,V)=" " AND J=V T
HEN  GOTO 8500
8070 IF J=V THEN  GOTO 8100
8080 IF A$(D)>=A$(D-V) THEN  GO
TO 8500
8090 LET X$=A$(D): LET A$(D)=A$
(D-V): LET A$(D-V)=X$: LET D=D-
V: GOTO 8020
8100 IF A$(D)<=A$(D+V) THEN  GO
TO 8500
8110 LET X$=A$(D): LET A$(D)=A$
(D+V): LET A$(D+V)=X$: LET D=D+
V: GOTO 8020
8500 SOUND .1,10: PRINT AT U,U;
"Registro numero ";D;"   ": RET
URN
9000 PRINT AT 19,U;"TEM CERTEZA
 DE QUE QUER APAGAR?": PAUSE U
9010 IF INKEY$="" THEN  GOTO 90
10
9020 IF INKEY$<>"S" THEN  PRINT
 AT 19,U;"
          ": SOUND .1,10: RETU
RN
9030 PRINT AT 19,U;" APAGANDO
                "
9035 LET DD=D
9040 IF D=R THEN  LET DD=DD-V:
GOTO 9060
9050 IF A$(D+V,V)<>" " THEN  LE
T A$(D)=A$(D+V): LET D=D+V: GOT
O 9040
9060 LET A$(D)="": FOR F=V TO 1
00: NEXT F: PRINT AT 19,U;"
      "
9070 LET D=DD: IF A$(V,V)=" " T
HEN  LET D=U: RETURN
9072 IF D=U THEN  LET D=V
9075 IF A$(D,V)=" " THEN  LET D
=D-V
9080 IF MD=V THEN  RETURN
9090 LET K=V
9100 IF A$(K,B(Z)+V TO B(Z+V))=
Z$(V) THEN  GOTO 9130
9110 IF K=R OR A$(K,V)=" " THEN
  LET DF=V: GOTO 4046
9120 LET K=K+V: GOTO 9100
9130 LET DD=D: LET PA=V
9140 IF A$(DD,V)=" " OR DD=U TH
EN  LET PA=2: LET DD=D: LET MO=
MO-V
9150 IF A$(DD,B(Z)+V TO B(Z+V))
=Z$(V) THEN  LET D=DD: RETURN
9160 LET DD=DD+MO: GOTO 9140
```

```
3000 GOSUB8500
3010 LOCATE0,22:PRINT"Modificar
 qual campo? (1 até ";A;")";:
3020 IN$=INKEY$:IFIN$=""THEN302
0
3030 J=VAL(IN$)
3040 IFJ<1ORJ>ATHEN3020
3060 LOCATE0,22:PRINTSPACE$(39)
;:LOCATE0,22:PRINT"Entre com o
campo modificado: ";:LINEINPUTA
$(D,J)
3070 A$(D,J)=LEFT$(A$(D,J),A(J)
)
3080 IFD=RTHENJ=-1:GOTO3130
3090 IFD=1THENJ=1:GOTO3120
3100 IFA$(D,1)>A$(D+1,1)THENJ=1
3110 IFA$(D,1)<A$(D-1,1)THENJ=-
1
3120 IFA$(D+1,1)=""ANDJ=1THEN31
80
3130 IFJ=1THEN3160
3140 IFA$(D,1)>=A$(D-1,1)THEN31
80
3150 FORN=1TOA:SWAPA$(D,N),A$(D
-1,N):NEXT:D=D-1:GOTO3080
3160 IFA$(D,1)<=A$(D+1,1)THEN31
80
3170 FORN=1TOA:SWAPA$(D,N),A$(D
+1,N):NEXT:D=D+1:GOTO3080
3180 FORF=1TO300:NEXT:RETURN
4000 G=D
4010 GOSUB8500
4020 LOCATE0,22:PRINT"Você conf
```

```
irma a deleção? (S/N)";
4030 IN$=INKEY$:IFIN$=""THEN403
0
4040 IFIN$<>"S"THENRETURN
4050 IFG=NRTHENG=G-1
4060 CLS:LOCATE12,15:PRINTCHR$(
7);"Apagando";CHR$(7);
4070 IFD=NRTHEN4090
4080 FORN=1TOA:A$(D,N)=A$(D+1,N
):NEXT:D=D+1:GOTO4070
4090 FORN=1TOA:A$(D,N)="":NEXT:
NR=NR-1:D=G
4100 FORF=1TO400:NEXT:RETURN
5000 FORN=1TOA:LOCATE5,2*N+2:PR
INTN,N$(N):NEXT
5010 LOCATE0,22:PRINT"Procurar
por qual campo? (1 até ";A;")";
5020 IN$=INKEY$:IFIN$=""THEN502
0
5030 IFVAL(IN$)<1ORVAL(IN$)>ATH
EN5020
5040 PRINTCHR$(7);:Z=VAL(IN$):L
OCATE0,22:PRINT"Procurar o que
no campo ";Z;"?";SPACE$(8)
5050 LINEINPUTZ$
5060 D=1:G=0:CH=1
5070 IFD>NRANDG=1THENG=0:CH=-1E
LSEIFD>NR1ЧEN5230
5080 IFD<1ANDG=1THENG=0:CH=1ELS
EIFD<1THEN5230
5090 IFA$(D,Z)<>Z$THEND=D+CH:GO
TO5070
5100 LD=D:G=1:GOSUB8500
5110 LOCATE3,22:PRINT"[P]rosseg
ue   [R]etorna   [M]enu   [C]o
rrige   [A]paga   [I]mprime"
;
5120 IN$=INKEY$:IFIN$=""THEN512
0
5130 IN=INSTR(1,R$,IN$)
5140 ONINGOTO5160,5160,5170,518
0,5190,5200,5210
5150 GOTO5120
5160 CH=1:D=D+1:GOTO5070
5170 CH=-1:D=D-1:GOTO5070
5180 RETURN
5190 GOSUB3000:GOTO5090
5200 GOSUB4000:GOTO5090
5210 GOSUB10000:GOTO5070
5220 GOTO5110
5230 CLS:LOCATE0,15:PRINT"Não e
ncontrei ";Z$;" em ";N$(Z)
5250 FORF=1TO500:NEXT:RETURN
```

```
3000  POKE 34,22: GOSUB 6500
3010  VTAB 23: CALL  - 958: PRI
NT "NUMERO DO CAMPO A SER MODIF
ICADO =>";: GET CP$
3020  IF CP$ < "0" OR CP$ > "8"
 THEN 3010
3030 CP =  VAL (CP$): IF CP = 0
 THEN  POKE 34,0: RETURN
3040  VTAB 23: HTAB 1: CALL  -
958
3050  PRINT "CAMPO CORRIGIDO =>
";: INPUT A$(D,CP)
3060 A$(D,CP) =  LEFT$ (A$(D,CP
),A(CP))
3070  IF CP <  > 1 THEN 3000
3080  IF A$(D,1) > A$(D + 1,1)
AND D <  > NR THEN V = 1: GOTO
3200
3090  IF A$(D,1) < A$(D - 1,1)
AND D > 0 THEN V =  - 1: GOTO 3
200
3100  GOTO 3000
3200  FOR N = 1 TO A:X$ = A$(D,
N):A$(D,N) = A$(D + V,N):A$(D +
 V,N) = X$: NEXT :D = D + V: GO
TO 3080
4000  GOSUB 6500: VTAB 23
4010  PRINT "CONFIRMA A DELECAO
? ";: GET IN$
4020  IF IN$ <  > "S" THEN  RET
URN
4030  IF D = NR THEN 4060
4040  FOR X = D TO NR
4050  FOR Y = 1 TO A:A$(X,Y) =
A$(X + 1,Y): NEXT : NEXT
4060 NR = NR - 1:D = NR: IF D =
 0 THEN  POP : RETURN
4070  RETURN
5000 DD = 1: PRINT : FOR X = 1
TO A: PRINT : HTAB 10: PRINT X;
":-";N$(X): NEXT
5010  VTAB 21: PRINT "PROCURAR
POR QUAL CAMPO? ";: GET CP$
5020 CO =  VAL (CP$): IF CO > A
 OR CO < 1 THEN  RETURN
5025  PRINT CO
5030  VTAB 22: HTAB 1: PRINT "P
ROCURAR O QUE EM ";N$(CO);: INP
UT PR$
5035  IF PR$ = "" THEN  RETURN
5040  HTAB 13: PRINT "PROCURAND
O...";
5050  FOR XX = DD TO NR: IF  LE
FT$ (A$(XX,CO), LEN (PR$)) = PR
$ THEN FL = 1:D = XX: GOSUB 602
0: GOTO 5200
5060  NEXT
5070  IF FL = 0 THEN  VTAB 21:
HTAB 1: CALL  - 958: PRINT "NAO
 ENCONTREI ";PR$;" EM ";N$(CO):
 FOR X = 1 TO 5000: NEXT : GOTO
 5090
5080  VTAB 21: HTAB 1: CALL  -
958: HTAB 8: PRINT "FIM DE ARQU
IVO ENCONTRADO": FOR X = 1 TO 5
000: NEXT
5090 FL = 0:PR$ = "": RETURN
5200  IF XX = NR THEN 5090
5210  VTAB 15: PRINT "CONTINUO
PROCURANDO ";PR$: PRINT " EM ";
N$(CO);"? (S/N) ";: GET IN$
5220  IF IN$ = "N" THEN 5090
5230  IF IN$ = "S" THEN DD = XX
 + 1: GOTO 5050
5240  GOTO 5210
```

PROGRAME JOGOS A CORES

O TRS-Color (e seus compatíveis nacionais, como o CP-400) possui cinco modalidades para elaboração de gráficos de alta resolução (**PMODE**), numerados de 0 a 4. Eles diferem na resolução, no grau de detalhe e nas cores dos desenho exibidos. **PMODE** é uma instrução BASIC, específica para esses micros, e significa *picture modes* (modalidades gráficas). Ela estabelece o conjunto de cores a ser empregado. Na modalidade de quatro cores — **PMODE** 1 ou 3 —, o conjunto 0 consiste de verde, amarelo, azul e vermelho, enquanto o conjunto 1 de cores é formado por branco, azul-piscina (ciano), magenta e laranja. Para o conjunto 0 de cores, digite o comando **SCREEN 1,0**; para o conjunto 1, recorra a **SCREEN 1,1**.

A diferença entre **PMODE 1** e **PMODE 3** se encontra na resolução da tela gráfica que eles utilizam. Resolução gráfica é o número de *pixels* (*picture element* ou elemento de figura) que podem ser traçados na tela e, como o nome diz, é a unidade mínima de construção de figuras. Ele corresponde a um bloco retangular "aceso" na tela.

O programa abaixo mostra a capacidade de trabalho com gráficos de resolução média do TRS-Color. Do jeito que está, ele roda apenas em máquinas *com gravador cassete*. Para executá-lo em uma máquina com Disk BASIC, altere os endereços nos comandos **POKE** nas linhas 30 e 50, mudando-os de 1800 e 1801 para 3584 e 3585, respectivamente.

```
10 PMODE3,1:PCLS:SCREEN1,0
20 FOR I=1 TO 9
30 READ A,B:POKE 1800+I*32,A:PO
KE 1801+I*32,B
40 NEXT
```

Você já sabe como construir blocos gráficos em preto e branco. Aprenda agora a colorir seus desenhos em micros da linha TRS-Color, empregando técnicas mais sofisticadas.

- COMO UTILIZAR CONJUNTOS DE CORES DIVERSAS
- COMO DEFINIR CORES EM BLOCOS GRÁFICOS
- EXPLORE OS **PMODE**

```
50 FOR K=10 TO 22:POKE 1800+K*3
2,192:POKE 1801+K*32,0:NEXT
60 GOTO 60
70 DATA 192,0,213,149,213,149,2
13,149,234,170,234,170,213,149,
213,149,213,149
```

O programa desenha uma bandeira e um mastro na tela. Os códigos dos blocos gráficos que constituem o desenho são especificados em uma linha **DATA**; o programa as lê e coloca os caracteres gráficos correspondentes na memória de vídeo, através do comando **POKE** ou **PRINT@**. Desta vez, para que você possa ver os blocos gráficos mais claramente, os códigos são colocados na página de memória de vídeo, a partir do endereço 1800, o que os situa no meio da tela. Isto ocorre porque a página gráfica usada (**PMODE 3**) tem seu endereço absoluto de início na locação 1536. O programa não funcionará em máquinas com BASIC de disco.

A linha 10 seleciona o **PMODE 3** (conjunto de cores 0) e limpa a tela. As linhas 20 a 40 lêem nove pares de dados que definem o desenho da bandeira, e os colocam diretamente na página de vídeo correspondente, através do comando **POKE**. O restante da bandeira é desenhado pela linha 50, que traça o mastro através de dois **POKE** (códigos 192 e 0), repetidos 22 vezes. A linha 60 forma um laço sem fim, cuja única função é "congelar" a tela gráfica. Pressionando <**BREAK**>, você interrompe o programa.

A principal diferença entre a técnica que tínhamos aprendido antes e a utilizada neste programa está nos dados de definição da figura. Os códigos estão em decimal e foram convertidos a partir de um número binário de oito bits. Mas, ao invés de bits individuais, que dizem ao computador para ligar ou desligar pixels individuais, os pares de bits estabelecem agora que cor será atribuída aos pares de pixels. No **PMODE 1**, um par de bits diz ao computador para situar dois pares de pixels, um abaixo do outro na tela.

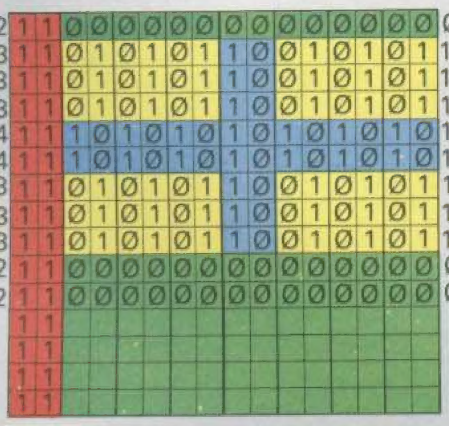
A figura 1 mostra como a bandeira é feita de pares de pixels: ela é definida através de um conjunto de números binários de dois bits, correspondentes a cada um dos dois blocos de pixels. No conjunto de cores 0 você deve escrever 00 para um bloco de cor verde, 01 para um bloco amarelo, 10 para um azul e 11 para um vermelho. Quatro desses números de dois bits compõem cada parte dos dados de oito bits.

Caso tenha escolhido o conjunto de cores 1, digitando **SCREEN 1,1** na linha 10, você terá uma bandeira desenhada em branco, ciano, magenta e laranja. Nesse conjunto de cores você escreve 00 quando quer um bloco branco opaco, 01 para um azul ciano, 10 para um bloco magenta e 11 para um laranja.

O programa abaixo permite ver o efeito que os códigos gráficos em **DATA** têm sobre os pixels e as suas cores em cada um dos **PMODE** (rodar apenas nos micros com BASIC para cassete).

```
10 CLS
20 INPUT"MODO GRAFICO(0-4) ";MO
30 MO=INT(MO):IF MO<0 OR MO>4 T
HEN 20
40 INPUT"NUMERO DA TELA(0-1) ";
ST
50 ST=INT(ST):IF ST<0 OR ST>1 T
HEN 40
60 INPUT "NUMERO(0-255) ";NU
70 NU=INT(NU):IF NU<0 OR NU>255
 THEN 60
80 IF MO<4 THEN LE=2 ELSE LE=1
90 IF MO<2 THEN DE=2 ELSE DE=1
100 IF(MO AND 1)=1 THEN SP=-2 E
LSE SP=-1
110 FOR K=7 TO 0 STEP SP
120 FOR L=1 TO LE
130 FOR J=1 TO DE
140 IF SP=-1 THEN PP=1358+32*J+
(3-K)*LE+L:CO=INT(.5+(NU AND 2^
K)/2^K):GOTO 160
150 PP=1393+32*J+L-K:CO=INT(.5+
(NU AND 3*2^(K-1))/2^(K-1))
160 POKE PP,113-SP*15+CO*(14-SP
)+ST*64
170 NEXT J,L
180 POKE PP-32*DE,47+K
190 POKE PP-32*DE+1+SP,48+K
200 IF SP=-2 THEN POKE PP+31,11
2-(CO>1)
210 POKE PP+21,112+(CO AND 1)
220 NEXT K
230 PRINT @482,"PRESSIONE QUALQ
UER TECLA PARA CONTINUAR";
240 SCREEN 0,1-ST
250 A$=INKEY$:IF A$="" THEN 250
260 GOTO 10
```

O programa lhe pedirá para selecionar uma modalidade de gráfico (**PMODE**) e o número de tela (**SCREEN**). A seguir, digite um número entre 0 e 255. Aparecerão na tela as cores e os tamanhos relativos dos pixels, com os números dos bits em cima e os seus equivalentes binários embaixo.



Nº DO PMODE	PIXELS	CONJUNTO DE CORES	
		SCREEN 1,0	SCREEN 1,1
0	▪▪	preto e verde	preto e branco
1	▪▪	verde, amarelo, azul e vermelho	branco, ciano, magenta e laranja
2	▪▪	preto e verde	preto e branco
3	▪▪	verde, amarelo, azul e vermelho	branco, ciano, magenta e laranja
4	▪	preto e verde	preto e verde

COMO ENTRAR CÓDIGO DE MÁQUINA

- COMO ENCONTRAR UM LUGAR PARA O CÓDIGO DE MÁQUINA
- PROGRAMA MONITOR
- SAIBA USAR O MONITOR
- RODE UM PROGRAMA

O micro responde automaticamente a comandos em BASIC. Nesta lição você aprenderá a trabalhar com código de máquina, usando um programa em BASIC e a instrução POKE.

Embora o código de máquina seja a linguagem de microprocessador, não existem meios de colocá-lo diretamente na memória de um computador pessoal. É preciso entrá-lo através de um outro programa — o que na maioria dos micros atuais significa utilizar a linguagem BASIC para esse programa, pois ela é a única disponível na memória ROM da máquina. Além disso, não podemos mandar o computador executar um programa em código de máquina utilizando um comando nesse código: é preciso utilizar também uma instrução em BASIC.

Existem programas especiais para quem deseja programar diretamente em linguagem de máquina (ou seja, utilizando códigos em hexadecimal). Esses programas são chamados de *monitores* e permitem, entre outras coisas, entrar um programa em hexadecimal na memória da máquina, executá-lo, listá-lo na tela ou na impressora, corrigi-lo, etc.

Os micros da linha TRS-80, compatíveis com as máquinas de Radio Shack americanas, modelos III e IV, têm um monitor disponível na memória. Esse monitor permite o trabalho direto com códigos hexadecimais e é bastante completo. Já os micros compatíveis com o Modelo I do TRS-80 não dispõem desse recurso, necessitando de um programa elaborado em BASIC, especial ou disponível comercialmente.

Os micros das linhas Apple II+, IIe e IIc, bem como o Microdigital TK-2000, em todas as suas versões, têm um programa monitor embutido, para trabalho direto com hexadecimais. O modelo TK-2000 conta ainda com uma vantagem adicional: um míni-Assembler disponível na ROM do equipamento, que permite a programação com códigos mnemônicos e que pode ser invocado diretamente do BASIC, pelo comando **MA**.

Essas máquinas não oferecem monitores ou montadores embutidos na ROM. Precisam, portanto, de programas externos, que devem ser carregados previamente na memória RAM, para o trabalho direto com a linguagem de máquina (códigos hexadecimais).

O instrumento básico para entrar uma rotina de máquina é o comando **POKE**. Ele serve para colocar os códigos de máquina, byte por byte, na memória do computador, usando *endereços absolutos* (isto é, o número real da locação de memória, em vez de um nome simbólico de variável, como o Assembler e o BASIC fazem).

Nos micros compatíveis com Sinclair Spectrum, ZX-81, e TRS-80 Modelo I (BASIC Nível II, da versão cassete), você não pode usar um hexadecimal com o comando **POKE**, apenas um número decimal. Assim, os hexadecimais de sua rotina de código de máquina devem ser convertidos para decimais, antes de serem entrados. Já o TRS-Color, os TRS-80 Mod. III e IV (com o Disk BASIC), o MSX e o Apple II aceitam hexadecimais no **POKE**.

ONDE COLOCAR O PROGRAMA

Antes de entrar um programa em código de máquina, você deve decidir onde irá colocá-lo na memória. Obviamente, não se deve pô-lo em uma área de memória utilizada pelo micro, senão o seu programa será destruído quando algum outro na memória ROM entrar em ação, ou o computador deixará de funcionar corretamente, pois terá algum dado alterado pelo seu programa.

Você também precisa ser cuidadoso quando utilizar as áreas de memória RAM normalmente reservadas pelo interpretador BASIC (essas áreas são descritas no manual de operação ou programação de seu micro). Como a rotina de código de máquina pode ser chamada de um programa em BASIC, se um código de máquina for gravado nessa área enquanto o programa em BASIC estiver sendo rodado, ocorrerão falhas de operação, perdas de programa, ou até o "congelamento" da máquina.

Você pode utilizar áreas de memória intermediária, como o *buffer* da interface para o gravador cassete ou para a impressora, se o programa for curto, e desde que você não esteja utilizando esses periféricos. Se você olhar o manual de programação do seu computador, verá também que alguns modelos têm na memória RAM áreas não usadas que podem ser empregadas para programas curtos em código de máquina.

Nos micros compatíveis com o Sinclair Spectrum, como o Microdigital TK-90X, você pode utilizar a área de gráficos definíveis pelo usuário (UDG) — entre FF58 e FFFF, no modelo de 48 K — desde que ela não esteja sendo utilizada pelo seu programa. Mas, normalmente, programas em código de máquina são colocados entre a área de gráficos UDG e o final da área BASIC, através da modificação do endereço do sistema, chamado RAMTOP (e que identifica o maior endereço disponível na memória RAM do computador), para um valor menor.

Desse modo, a área para o BASIC fica diminuída, mas, como não pode ultrapassar o endereço armazenado na locação RAMTOP, o código de máquina é protegido contra qualquer interferência por parte do BASIC.

A RAMTOP é uma variável de sistema, e sua posição é fornecida pelo apontador nas locações de memória 5CB2 e 5CB3. Para movimentar o RAMTOP para baixo você precisa apenas colocar, com o comando **POKE**, um valor mais baixo nessas duas locações. Mas o Spectrum possui um outro comando BASIC: o **CLEAR**, que faz essencialmente a mesma coisa e deve ser seguido pelo endereço decimal do lugar para o qual você quer que o RAMTOP seja rebaixado.

O comando **CLEAR** também limpa o arquivo de exibição (ou seja, a tela), da mesma forma que um **CLS** (e todas as va-

RAMTOP
BASIC
A0 00
85 FE
A8 00
85 FC
A9 04
85 FB
A9 00
A1 08
00
35 35
32
30
36

riáveis). Além disso, ele restaura a posição do comando gráfico **PLOT** para as coordenadas 0,0 — que definem o lado inferior esquerdo da tela —, restabelece o apontador do comando **READ** para o início da lista de declarações **DATA** do programa, e zera todos os **RETURN** ativos. Em um TK-90X com 16 K, por exemplo, o comando mais usual é:

```
CLEAR 31999
```

Isso modifica o RAMTOP para 31 999 ou 7CFF em hexa, deixando 600 bytes acima dele — isto é, protegendo a área que vai desde esse ponto até o final da área de UDG (que está em 32 600) para a colocação de programas em código de máquina. No modelo de 48 K, o comando deve ser, neste exemplo:

```
CLEAR 63999
```

O RAMTOP desce, de tal modo que a área do BASIC passa a terminar em 63 999, ou F999 em hexa: você pode, assim, iniciar o seu programa de código de máquina em 64 000 ou FADO em hexa. Na maioria das aplicações isso deixará espaço suficiente para os seus programas em código de máquina.

Com programas de código de máquina mais longos que o usual, você deve abaixar o RAMTOP ainda mais. Mas isso pode não ser bom na prática: com o RAMTOP tão baixo, não existe espaço nem mesmo para entrar uma linha de BASIC.

S

No ZX-81, pode-se abaixar o RAMTOP, alterando as locações de memória 16 388 e 16 389 por meio de comandos **POKE**, de modo a criar uma área protegida, onde os programas de código de máquina não sejam modificados. O problema é que não se pode armazenar em fita o programa que estiver nessa área.

Programas curtos em código de máquina podem ser colocados com **POKE** na memória intermediária (*buffer*) para a impressora, que ocupa as locações de memória de 16 444 a 16 476. Existem outras áreas pequenas nas quais você pode colocar seus programas em código de máquina, mas não é possível guardá-los em fita.

Existem três maneiras de se colocar um programa de código de máquina em uma área BASIC, protegendo-o contra outras gravações: pode-se entrá-lo como parte de declarações **REM**; colocá-lo dentro de variáveis alfanuméricas ou cordões de caracteres; ou pode-se armazená-lo em um conjunto numérico.

Para entrar um programa em código de máquina como parte de uma declaração **REM** você deve calcular qual será sua extensão (contar os bytes). Então, na primeira linha do programa em BASIC, digite um **REM** seguido de pontos:

```
10 REM .....................
```

O número de pontos deve ser pelo menos igual ao número de bytes em seu programa de código de máquina.

Em seguida, você deve colocar os códigos do seu programa nas locações de memória ocupadas pela declaração **REM**, um byte de cada vez. Isto é feito por intermédio da declaração **POKE**. Se sua **REM** estiver na primeira linha do programa em BASIC, a área reservada para isso começa na locação de memória 16 514.

Outra maneira de criar um espaço protegido é dimensionar um conjunto. A linha de programa em BASIC:

```
10 DIM A(100)
```

fornecerá 500 locações livres da memória na área de variáveis. Para cada elemento desse conjunto, o ZX-81 deixa cinco locações livres. Se você quiser descobrir onde começar a armazenar o código de máquina neste caso, deverá utilizar a função **PEEK**, para obter o valor armazenado na variável de sistema **VARS**, que está localizada em 16 400 e 16 401:

```
PRINT PEEK 16400+PEEK 16401+6
```

Isso fornece o endereço inicial da área protegida que você criou (as seis locações extras de memória contêm detalhes do conjunto). Você poderá, então, usar a instrução POKE para entrar o programa em código de máquina, a partir do endereço reservado. É possível também criar espaço em um cordão com uma linha BASIC, como:

```
10 LET S$=".....................
```

com tantos pontos quanto bytes no programa de código de máquina. Para achar o endereço inicial, você deve examinar o que tem em **VARS** e acrescentar 6. As seis locações extras de memória, neste caso, contêm o nome do cordão.

O problema com esses dois métodos é que a área das variáveis é apagada quando se aperta a tecla **CLEAR** ou se roda o programa em BASIC de novo.

T

O modo mais fácil de proteger os programas de código de máquina nesse computador consiste em utilizar o comando **CLEAR** do BASIC. O **CLEAR** também zera todas as variáveis numéricas alfanuméricas. Usualmente, ele assume esta forma:

```
CLEAR 200,30000
```

Ele limita o topo da memória RAM (normalmente, em H7FFF ou 32 767 em decimal) em 30 000, deixando 2 767 bytes livres para os programas de código de máquina.

Nesses micros o comando **CLEAR** tem outra função separada, e não relacionada: a atribuição de espaço de memória para as variáveis alfanuméricas. O primeiro número do comando **CLEAR** fornece o espaço de cordões que deve ser reservado para o programa em BASIC. Quando a máquina é ligada, o computador guarda 200 bytes logo abaixo do endereço máximo de memória (RAMTOP) paras as variáveis alfanuméricas. Assim, quando você deixar o RAMTOP mais baixo, deverá reservar, abaixo dele, também uma quantidade similar (200 bytes), para os cordões.

Teoricamente, você pode baixar o RAMTOP do TRS-Color até quase o topo da área reservada para as telas gráficas, embora a máquina vá precisar de um mínimo de espaço para o BASIC.

O **PCLEAR 1** deixa apenas uma das páginas de gráficos; assim, você usa o **CLEAR** para limitar a memória em §H3680 (em computadores sem disco, apenas). O **PCLEAR 8** atribui as oito páginas de gráficos, permitindo que se reserve memória só até §H3680. Se você não utiliza o comando **CLEAR**, o computador atribui quatro páginas de gráficos; assim, você só poderá usar o **CLEAR** até §H1E80.

Esses limites dependem muito do que está dentro da máquina no momento. Se existir algum tipo de programa em BASIC, provavelmente você obterá uma mensagem de erro "OM" (*out of memory*). Assim, para todos os propósitos práticos, trabalhar próximo ao limite não será bom, pois você deverá entrar um programa BASIC em algum lugar para chamar o seu programa de código de máquina. Se você quiser iniciar seu programa de código de máquina com um número redondo como 30 000 então digite:

```
CLEAR 200,29999
```

N

Para proteger os programas de código de máquina neste micro utilize o comando **CLEAR** do BASIC. Este limita a memória RAM máxima, reservando um espaço acima para seu programa em código de máquina. O **CLEAR** também zera as variáveis numéricas alfanuméricas. Normalmente, ele assume esta forma:

```
CLEAR 100,&HE000
```

Ele limita o topo da memória RAM em 57 344, deixando 8 192 bytes livres para os

AMTOP
1FF8 55 59
1FF7 2A
1FF6 4F
1FF5 4B
1FF2 2E 4D 4B
1FEF 2C 2F 2C
4B 3A
3A
4B
3A
4B
4C 3A
FE4 4B
FE1 4C 4C
FE0 4B

programas de código de máquina.

Nestes micros o comando **CLEAR** tem uma outra função separada e não relacionada: a atribuição de espaço de memória para as variáveis alfanuméricas. O primeiro número do comando **CLEAR** fornece o espaço de cordões que deve ser reservado para o programa em BASIC. O computador guarda automaticamente 100 bytes abaixo do endereço máximo de memória (RAMTOP) para as variáveis alfanuméricas quando a máquina é ligada. Se você deixar o RAMTOP artificialmente mais baixo, deverá destinar também uma quantidade similar — 100 bytes — abaixo dele, para os strings.

UM PROGRAMA MONITOR

O programa seguinte é um monitor simples para auxílio à programação em linguagem de máquina. Ele permite que você digite códigos de máquina na memória do computador, guarde programas de código de máquina em fita e examine-os na memória. Antes de entrar o programa, é necessário dar o comando **CLEAR** apropriado. Feito isso, o programa é entrado e rodado; em seguida, o computador pede que você forneça um endereço inicial, que deverá ser o da memória, acima do estabelecido em **CLEAR**. Lembre-se de que o comando **CLEAR** especifica o último endereço de BASIC, e o seu programa de código de máquina deverá começar na locação acima dele.

Agora digite o monitor de código de máquina para o seu computador.

```
10 CLS
20 PRINT @193,"1: ENTRAR EM COD
IGO DE MAQUINA "
30 PRINT @257,"2: EXAMINAR MEMO
RIA "
40 PRINT @321,"3: GRAVAR BYTES
NA FITA "
50 A$=INKEY$:IF A$<"1" OR A$>"3
" THEN GOTO 50
60 CLS
70 ON VAL(A$) GOSUB 200,400,600
80 GOTO 10
200 INPUT "ENDERECO INICIAL ";S
A
210 LINE INPUT D$
220 IF D$="" THEN GOTO 210
230 IF LEFT$(D$,1)="#" TTHEN RE
TURN
240 S$=LEFT$(D$,2)
250 POKE SA,VAL("&H"+S$)
260 PRINT SA,LEFT$(D$,2)
270 D$=MID$(D$,3)
280 SA=SA+1: GOTO  220
400 INPUT"ENDERECO INICIAL ";SA
410 PRINT"SAIDA PARA IMPRESSORA
?(S/N)"
420 A$=INKEY$:IF A$="" THEN 420
430 P=0:IF A$="S" THEN P=-2
440 PRINT #P,SA;
450 FOR M=0 TO 7
460 PRINT #P," ";RIGHT$(" "+HEX
$(PEEK(SA+M)),2);
470 NEXT:PRINT#P
480 SA=SA+8
490 A$=INKEY$:IF A$="" THEN 490
500 IF A$=CHR$(13) THEN RETURN
510 GOTO 440
600 INPUT "ENDERECO INICIAL ";S
A
610 INPUT "NUMERO DE BYTES ";N
620 LINE INPUT "NOME DO ARQUIVO
 ";N$
630 CSAVEM N$,SA,SA+N,SA
640 RETURN
```

```
  5  POKE 23658,8
 10 PRINT  INVERSE 1;AT 5,6;"
MONITOR  TK 90 X "
 15 PRINT AT 8,1;"1: ENTRAR EM
  CODIGO DE MAQUINA"
 20 PRINT AT 10,1;"2: EXAMINAR
  MEMORIA"
 30 PRINT AT 12,1;"3: GRAVAR B
YTES EM FITA"
 70 LET A$=INKEY$: IF A$<"1"
OR A$>"3" THEN  GOTO 70
 80 CLS : GOSUB 100+200*(VAL A
$-1)
 90 RUN
100 INPUT "ENDERECO INICIAL? "
;SA
110 INPUT  LINE D$
120 IF D$="" THEN  GOTO 110
125 IF D$(1)="#" THEN  RETURN
127 IF LEN D$=1 THEN  GOTO 110
130 LET S$=D$(1 TO 2)
140 FOR N=1 TO 2
150 IF S$(N)>"9" THEN  LET S$(
N)=CHR$ (CODE S$(N)-7)
155 IF S$(N)>"?" OR S$(N)<"0"
THEN  PRINT  FLASH 1;"CARACTER
E INVALIDO": GOTO 110
160 NEXT N
170 POKE SA,(CODE S$(1)-48)*16
+CODE S$(2)-48
180 PRINT SA,D$( TO 2)
190 LET D$=D$(3 TO )
200 LET SA=SA+1
210 GOTO 120
300 INPUT "ENDERECO INICIAL? "
;SA
310 INPUT "IMPRESSORA (S/N)? "
; LINE P$
320 LET ST-2: IF P$-"S" THEN
LET ST=3
330 PRINT #ST;SA;
340 FOR M=0 TO 7
350 LET H$="##"
360 LET H$(1)=CHR$ (INT (PEEK
(SA+M)/16)+48)
370 LET H$(2)=CHR$ (48+PEEK (S
A+M)-16*(CODE H$(1)-48))
380 FOR N=1 TO 2
390 IF H$(N)>"9" THEN  LET H$(
N)=CHR$ (CODE H$(N)+7)
400 NEXT N
410 PRINT #ST;TAB 7+3*M;H$;
420 NEXT M
440 LET SA=SA+8
445 PRINT #ST: POKE 23692,0
450 LET A$=INKEY$: IF A$=""
THEN  GOTO 450
460 IF A$=CHR$ 13 THEN  RETURN

470 GOTO 330
500 INPUT "ENDERECO INICIAL? "
;SA
510 INPUT "NUMERO DE BYTES? ";
N
520 INPUT "NOME DO ARQUIVO? ";
LINE N$
530 IF LEN N$<1 OR LEN N$>10
THEN  GOTO 520
540 SAVE N$CODE SA,N
550 RETURN
```

Antes de executar este programa, você deve digitar uma linha 1 contendo um **REM** seguido do número apropriado de caracteres, de modo a receber seu programa em código de máquina (um caractere por byte de programa).

```
 10 PRINT AT 4,9;"MONITOR TK85"
 15 PRINT AT 8,4;"1 - PROGRAMAR
  EM CODIGO"
 20 PRINT AT 10,4;"2 - EXAMINAR
  A MEMORIA"
 50 LET A$=INKEY$
 60 IF A$<"1" OR A$>"2" THEN
 GOTO 50
 70 CLS
 80 GOSUB 100+200*(VAL A$-1)
 85 CLS
 90 RUN
100 PRINT AT 21,0;"ENDERECO INI
CIAL?"
105 INPUT SA
110 INPUT D$
120 IF D$="" THEN GOTO 110
125 IF D$(1)="$" THEN RETURN
130 LET S$=D$(1 TO 2)
135 SCROLL
140 FOR N=1 TO 2
150 IF S$(N)<"0" OR S$(N)>"F"
THEN PRINT "CARACTER INVALIDO"
155 IF S$(N)<"0" OR S$(N)>"F"
THEN GOTO 110
160 NEXT N
170 POKE SA,(CODE S$(1)-28)*16+
CODE S$(2)-28
175 SCROLL
180 PRINT SA,D$( TO 2)
190 LET D$=D$(3 TO )
195 IF LEN D$=1 THEN GOTO 110
200 LET SA=SA+1
210 GOTO 120
300 PRINT AT 21,0;"ENDERECO INI
CIAL?"
310 INPUT SA
320 SCROLL
330 PRINT SA;
340 FOR M=0 TO 7
350 LET H$="  "
360 LET H$(1)=CHR$ (INT (PEEK (
SA+M)/16)+28)
370 LET H$(2)=CHR$ (28+PEEK (SA
```

```
+M)-16*(CODE H$(1)-28))
400 PRINT TAB 7+3*M;H$;
410 NEXT M
440 LET SA=SA+8
450 LET A$=INKEY$
460 IF A$="" THEN GOTO 450
470 IF A$=CHR$ 118 THEN RETURN
480 GOTO 320
```

```
10 CLS
20 LOCATE 8,7:PRINT "1 - Progra
mar em código"
30 LOCATE 8,9:PRINT "2 - Examin
ar memória"
40 LOCATE 8,11:PRINT "3 - Grava
r bytes na fita"
50 A$=INKEY$:IF A$="" THEN GOTO
 50
60 CLS
70 ON VAL(A$) GOSUB 200,400,600
80 GOTO 10
200 INPUT "Endereço inicial";SA
210 LINE INPUT D$
220 IF D$="" THEN GOTO 210
230 IF LEFT$(D$,1)="$" THEN RET
URN
240 S$=LEFT$(D$,2)
250 POKE SA,VAL(&H0+S$)
260 PRINT SA,LEFT$(D$,2)
270 D$=MID$(D$,3)
280 SA=SA+1:GOTO 220
400 INPUT "Endereço inicial";SA
410 PRINT "Saída para impressor
a ? (S/N)"
420 A$=INKEY$:IF A$="" THEN 420
430 IF A$="S" OR A$="s" THEN OP
EN "LPT:" FOR OUTPUT AS #1 ELSE
 OPEN "CRT:"FOR OUTPUT AS #1
440 PRINT #1,SA;
450 FOR M=0 TO 7
460 PRINT #1," ";RIGHT$(" "+HEX
$(PEEK(SA+M)),2);
470 NEXT:PRINT #1,
480 SA=SA+1
490 A$=INKEY$:IF A$="" THEN 490
500 IF A$=CHR$(13) THEN CLOSE #
1:RETURN
510 GOTO 440
600 INPUT "Endereço inicial ";S
A
610 INPUT "Número de bytes ";N
620 LINE INPUT "Nome do arquivo
 ";N$
630 X$="CAS:"+N$
640 BSAVE X$,SA,SA+N,SA
650 RETURN
```

COMO O MONITOR FUNCIONA

Quando você rodar o programa apropriado à sua máquina, ele exibirá um menu onde aparecem as três opções básicas de trabalho com o monitor: "Entrar Código de Máquina", "Examinar a Memória" ou — exceto no ZX-81 — "Armazenar Bytes na Fita".

Se você quiser entrar código de máquina, deverá pressionar a tecla 1. Depois de fornecer um endereço inicial para a máquina e pressionar <**ENTER**> ou <**RETURN**>, você deve começar a digitar os códigos de máquina na forma de valores hexadecimais de dois dígitos cada. Cada valor deve ser separado do seguinte por um espaço em branco.

Antes de pressionar <**ENTER**> ou <**RETURN**> você pode entrar tantos pares quantos desejar. Mas é melhor digitar uma linha por vez e checar rigorosamente os dígitos antes de pressionar <**ENTER**>. É muito mais fácil editá-los enquanto eles ainda estão na tela do que quando estiverem na memória.

Você notará que as versões do programa para o Spectrum e para o ZX-81 traduzem seus números hexa para decimais, antes de colocá-los na memória. Isto deve ser feito, como explicamos antes, porque o **POKE** é uma declaração do BASIC, e o BASIC nestas duas máquinas não aceita números hexa. Mesmo assim, é melhor você pensar apenas em hexa pois isso lhe dará uma melhor percepção de como o computador funciona.

Você deve finalizar os seus programas de código de máquina com um sinal sustenido # ou, no ZX-81, um sinal de dólar. Isso trará o menu de volta para a tela.

Se você selecionar a opção 2 para examinar a memória, o programa pedirá novamente um endereço inicial — desta vez é o início da área de memória que precisa ser observada. Para examinar todo o seu programa em código de máquina, esse endereço inicial deverá ser o mesmo que o endereço inicial que você forneceu anteriormente.

A seguir, exceto no ZX-81, será perguntado se você quer imprimir uma cópia definitiva do programa em código de máquina na sua impressora. Se você não quiser fazer isso, e pressionar N, o programa imprimirá o endereço inicial e os conteúdos dessa locação e das sete locações de memória subseqüentes em uma linha na tela.

Se, ao contrário, você responder à questão acima pressionando qualquer tecla do teclado — exceto <**ENTER**> ou <**RETURN**> —, o endereço original será impresso, junto com seus conteúdos e os conteúdos dos sete bytes subseqüentes. Assim, seguindo esse método, você poderá imprimir todo o seu programa de código de máquina.

Se você localizar um erro e quiser corrigi-lo, pressione <**ENTER**> ou <**RETURN**>, e o programa retornará ao menu. Pressione 1 para entrar o código de máquina, e o computador pedirá novamente um endereço inicial. Desta vez, forneça o endereço do byte que estiver errado. Se o erro atingir toda uma série, forneça apenas o endereço do primeiro byte da série; em seguida, entre os bytes corretos.

Outro modo de fazer correções, quando existe apenas um erro, consiste em acionar a tecla <**BREAK**> (nos micros da linha Sinclair e TRS-Color), ou as teclas <**CONTROL**> <**STOP**> (no MSX), para interromper o programa monitor, e usar o **POKE** de modo direto, para corrigir o conteúdo da locação apropriada. Lembre-se de que no Spectrum e no ZX-81 você deve usar o **POKE** com um número decimal. Assim que terminar a correção, pressione <**ENTER**> ou <**RETURN**> e examine mais uma vez a memória para ter certeza de que dessa vez você acertou.

GUARDE SUAS ROTINAS

Se você quiser conservar o monitor de código de máquina, deverá guardá-lo em uma fita separado de seus programas de código de máquina, da maneira usual. A opção "Armazenar Bytes em Fita" no menu do programa monitor serve apenas para guardar as rotinas de código de máquina que foram entradas com o monitor. A única exceção é para o ZX-81, no qual você deve guardar suas rotinas de código de máquina junto com o programa monitor.

Nas outras máquinas, se você pressionar 3, o programa pedirá mais uma vez um endereço inicial. Este deverá ser normalmente o endereço inicial da rotina de máquina que você acabou de entrar, embora qualquer parte da memória possa ser guardada por meio da utilização de um endereço diferente.

As versões deste programa para o Spectrum, o TRS-Color e o MSX pedirão então a você os números de bytes que a sua rotina em código de máquina ocupa. Você pode calcular isso pela contagem dos pares de dígitos hexas que foram entrados anteriormente. Cada par é um byte. Em seguida, você deve entrar um nome para a rotina — assim, o seu computador será capaz de identificá-lo quando você quiser carregá-lo. A rotina será gravada pressionando-se as teclas <**PLAY**> e <**RECORD**> no gravador, antes de acionar a tecla <**ENTER**> do computador. Quando a gravação se completar, aparecerá o menu na tela, novamente.

COMO CARREGAR CÓDIGO DE MÁQUINA

Com o Spectrum, o TRS-Color e o MSX a rotina de código de máquina é carregada de modo normal (por exemplo, no MSX deve-se utilizar o comando **BLOAD "CAS:ROTINA"**, onde

ROTINA é o nome dado à rotina quando esta foi gravada). Contudo, se o monitor na memória da máquina estiver rodando, você deverá antes disso pressionar <**BREAK**> ou <**CONTROL**> <**STOP**>, para interrompê-lo.

Lembre-se de repetir **CLEAR** para proteger uma área de memória para o programa em código de máquina, caso você tenha desligado a máquina, ou alterado a posição do RAMTOP desde que o digitou.

Com o ZX-81, o seu programa em código de máquina será automaticamente carregado da fita junto com o monitor.

ROLE PARA TRÁS

Para exemplificar o uso do monitor, tente o programa listado abaixo. Com exceção dos micros da linha MSX, esse programa rola a tela para trás — isto é, de cima para baixo e não o contrário. Para o MSX, a rotina apresentada serve para preencher a tela com traços coloridos (é o mesmo programa apresentado na primeira lição desta série). Entre a rotina adequada à sua máquina, utilizando o programa monitor. Termine digitando um sinal #.

T

```
8E 05 E0 A6 84 A7 89 FE
00 30 88 E0 A6 84 A7 88
20 8C 04 00 2C F3 30 89
02 01 8C 06 00 25 E4 39
#
```

S

```
21 9F 58 11 BF 58 06 08
25 15 E5 D5 C5 01 A0 00
ED B8 06 02 C5 D5 11 00
F9 19 D1 01 20 00 ED B8
E5 21 00 F9 19 EB E1 01
E0 00 EB B8 C1 10 E5 C1
D1 E1 10 D4 21 00 3F 06
08 C5 24 E5 AF 77 54 5D
13 01 1F 00 ED B0 E1 C1
10 EF 21 9F 5A 11 BF 5A
01 A0 02 ED B8 21 00 58
11 01 58 3A 8D 5C 77 01
1F 00 ED B0 C9 #
```

SS

```
01 18 03 2A DC 40 09 54
5D 01 F7 02 2A DC 40 09
ED B8 2A DC 40 23 36 00
54 5D 13 01 1F 00 ED B0
C9 $
```

TS

```
98 ED 6B 02 03 3E 19 06
FF ED B3 3D 20 FA C9 02
18 F9 C9 #
```

RODE OS PROGRAMAS

Uma vez digitado o programa em código de máquina e verificados os possíveis erros, chegou o momento de rodá-lo. Lembre-se que ele não reagirá ao comando **RUN** normal. Para rodar programas em código de máquina, são necessárias instruções especiais que variam de computador para computador.

S

O Spectrum e o ZX-81 utilizam a função **USR** (abreviatura de USer Routine) do BASIC, seguida de um endereço inicial do programa em código de máquina. O **USR** faz retornar o valor decimal contido no registro interno BC do microprocessador, uma vez que se tenha completado a execução da rotina em código de máquina. Isto não é muito útil para você, no momento, mas significa que se esse retorno não ocorrer normalmente o programa em código de máquina não deve ter rodado corretamente.

O **USR** não é um comando, mas sim uma função. E a estrutura do BASIC do Sinclair exige que uma linha executada seja iniciada com um comando. Assim, o **USR** deve ser prefaciado por uma palavra de comando. Se você utilizou os endereços iniciais sugeridos neste artigo, para o Spectrum, faça:

```
RANDOMIZE USR 32000
```

e no ZX-81 utilize:

```
RAND USR 16514
```

(observe que a tecla **RAND** no Spectrum fornece **RANDOMIZE** na tela. No ZX-81 a tecla **RAMD** fornece **RAND** na tela). Em si próprio, o comando não tem sentido, mas ele dá resultado quando você rodar a rotina em código de máquina. Lembre-se de que a linha acima é apenas um exemplo. Troque o número que vem depois de **USR** pelo endereço decimal inicial de seu programa, fornecido para o monitor quando digitado.

Na verdade, qualquer comando BASIC pode ser utilizado para prefaciar **USR**. Por exemplo, o **PRINT USR 32000**, embora este, na verdade, imprima o valor do par de registros BC na tela, e isso sempre seja desejável.

RANDOMIZE USR 32000 ou **RAND USR 16514)** é mais utilizado porque evita efeitos colaterais indesejados. Mas deverá ser evitado se você estiver utilizando números aleatórios em qualquer parte do programa, pois ele irá acionar o gerador de números aleatórios novamente. Então, utilize no Spectrum:

```
LET L = USR 32000
```

e no ZX-81 empregue:

```
LET L = USR 16514
```

Isso coloca o valor dos registros BC na variável **L**, quando o programa em código de máquina termina de rodar — isso pode não ter nenhuma utilidade para o seu programa, mas no Sinclair os comandos **LET**, **L**, = e **USR** estão todos nas mesmas teclas, o que simplifica o trabalho de digitar.

T

O TRS-Color também possui instruções para executar programas em código de máquina. **EXEC** é um desses comandos; deve ser seguido pelo endereço inicial da rotina em código de máquina que você quiser rodar. Por exemplo:

```
EXEC 24000
```

rodará a rotina do código de máquina que se inicia na locação de memória em 24000 decimal. Existe também a função **USR**, que aceita parâmetros ou variáveis para serem passados entre o programa BASIC e a rotina em código de máquina. Antes de começar, você precisa definir onde deve se iniciar a rotina em código de máquina. Isso deve ser feito por meio da instrução **DEFUSR**, seguida de um número de 0 a 9. Assim, você pode definir até 10 rotinas **USR** diferentes dentro de um programa em BASIC. Quando **DEFUSR** não é seguido de um número o computador estabelece que seu significado equivale a **DEFUSR 0**. A sintaxe normal em um programa BASIC é:

```
10 DEFUSR1 = 24000
```

Isso define a rotina **USR** número 1, como se iniciando na locação 24000 decimal da memória.

A função que realmente executa o código de máquina é **USR**. Devido a um defeito na ROM, para chamar uma rotina de código de máquina do TRS-Color, **USR** deve ser seguida de 0 e o número da rotina que você já definiu.

A sintaxe normal é:

```
P = USR01(Q) ou PRINT USR01(Q)
```

A função **USR** copia o valor da variável **Q** no acumulador interno do microprocessador. Assim, ela pode ser captada e usada pela rotina em código de máquina, se for necessário.

Neste exemplo, quando a rotina em código de máquina termina sua execução, os conteúdos do acumulador retornam ao programa BASIC, como variável **P**.

Como em outros casos, variáveis de

string também podem ser transmitidas entre o programa em BASIC e programas em código de máquina.

Para executar programas em código de máquina, o MSX utiliza a função do BASIC chamada **USR** (abreviatura do USer Routine), seguida de um endereço inicial do programa de código de máquina. A **USR** retorna o valor decimal contido em um par de registros internos do microprocessador, uma vez que se tenha completado a execução da rotina em código de máquina. Isto não é muito útil para você, no momento, mas, se este retorno não ocorrer normalmente, significa que o programa em código de máquina não deve ter rodado corretamente.

A função **USR** aceita também parâmetros ou variáveis para serem passados entre o programa BASIC e a rotina em código de máquina.

Como já foi dito anteriormente, **USR** é uma função e não um comando. Ora, a estrutura do BASIC do MSX impõe que ela esteja contida numa linha iniciada com um comando. Podemos utilizar então algo do gênero:

```
P = USR(Q) ou PRINT USR(Q)
```

A função **USR** copia o valor da variável **Q** no acumulador interno do microprocessador. Assim, ele pode ser captado e usado pela rotina em código de máquina, em caso de necessidade.

Tal como nos exemplos já vistos, quando a rotina em código de máquina termina sua execução, os conteúdos do acumulador retornam ao programa BASIC, como a variável **P**. Para a rotina apresentada neste programa, é necessário preparar a tela gráfica antes de chamá-la, pois ela não faz isso internamente. Para executá-la, faça o seguinte programa e rode-o com o comando **RUN**:

```
1 SCREEN 2
2 PSET(0,0)
3 A=USR(B)
4 GOTO 4
```

Como foi reservado o espaço em RAM acima de &HE000, usamos o endereço inicial - 8191 ou §HE001, em hexadecimal, de modo a iniciar o **USR**. O número em decimal é negativo pois é o complemento do número decimal que normalmente apareceria como negativo.

Uma observação final para a rotina de exemplo para o **MSX**: ela usa a tela gráfica (**SCREEN1**); por isso, se o leitor acidentalmente rodar a rotina sem o programa dado acima (que retorna as coisas ao normal), deverá digitar **SCREEN 0** e pressionar a tecla **<RETURN>** depois.

O QUE SÃO VARIÁVEIS

Todos aqueles Xs e Ys esquisitos nos programas tornam-se simples quando se entende realmente sua função. Aprenda como trabalhar com variáveis em seus próprios programas.

Quando você quiser armazenar uma informação qualquer na memória do computador, será necessário, identificá-la; do contrário, o computador será capaz de descobrir onde está a informação no momento em que você precisar dela novamente. Muitas vezes a informação que se pretende armazenar consiste em um número cujo valor se modifica sempre que o computador executa uma repetição (laço) em uma seção do programa. Abaixo temos um exemplo de uma situação desse tipo:

```
10 LET X=0
20 LET X=X+1
30 PRINT X;" ";
40 FOR T=0 TO 10
50 NEXT T
60 GOTO 20
```

A linha 60 cria o laço, fazendo com que o programa se repita indefinidamente. O número representado por X (originalmente zerado pela linha 10) atinge a linha 20 e é aumentado em 1. O X é chamado de variável numérica, pois armazena um número.

Para entender melhor como funciona uma variável numérica, imaginemos que a memória do computador é constituída por uma série de caixas ou compartimentos. Cada caixa recebe um nome. Num computador simples, esses nomes poderão ser letras do alfabeto. Para armazenar um número na memória, bastará atribuir esse valor a uma das caixas:

```
LET C = 25
```

Essa instrução armazena o valor 25 na variável C. Nem todos os computadores exigem o emprego do comando **LET**, mas é uma boa idéia incluí-lo até que você esteja experiente.

Uma vez armazenado, esse valor pode ser utilizado por outras declarações em outros cálculos, etc. Você pode digitar, por exemplo:

```
PRINT C * 4
```

ou:

```
PRINT C/5
```

Essas declarações deverão fornecer as respostas 100 e 5 respectivamente, se continuarmos com o valor definido para **C**, acima. Mas se você digitar:

```
PRINT C
```

verá que o valor de **C** permanece inalterado. Para modificar o valor de **C**, aumentando-o para 30, por exemplo, digite diretamente:

```
LET C = 30
```

ou, ainda:

```
LET C = C + 5
```

Note que a linha de comando acima não faz sentido do ponto de vista matemático. Entretanto, o comando **LET** não deve ser entendido como uma equação, apesar do sinal de igualdade. A declaração acima significa apenas: "tome o valor numérico armazenado na variável **C**, some 5 a ele, e coloque de volta o resultado na variável **C**". O que acontece é que o valor anterior presente em **C** (no caso, 25) é apagado quando armazenamos um novo valor na variável.

OS NOMES DAS VARIÁVEIS

A combinação de caracteres permitidos pelo BASIC para se dar nomes às variáveis numéricas difere de computador para computador. As principais regras são mostradas no box na página 99.

Mas uma coisa que nenhum computador aceitará é um nome de variável que se inicie por um número. Por exemplo:

```
LET 7 = 14
```

não tem sentido. Um nome de variável pode incluir números (exemplos: VAR7, A1985, etc.), mas o primeiro caractere precisa ser sempre uma letra. Além disso, não é permitido utilizar como nome de variáveis palavras que representem comandos, como **TO, THEN, OR** ou **AND**, etc. Alguns micros não aceitam que essas palavras-chaves façam parte do nome da variável: a variável BEND, não seria aceita pelo interpretador BASIC, pois inclui a expressão **END.**

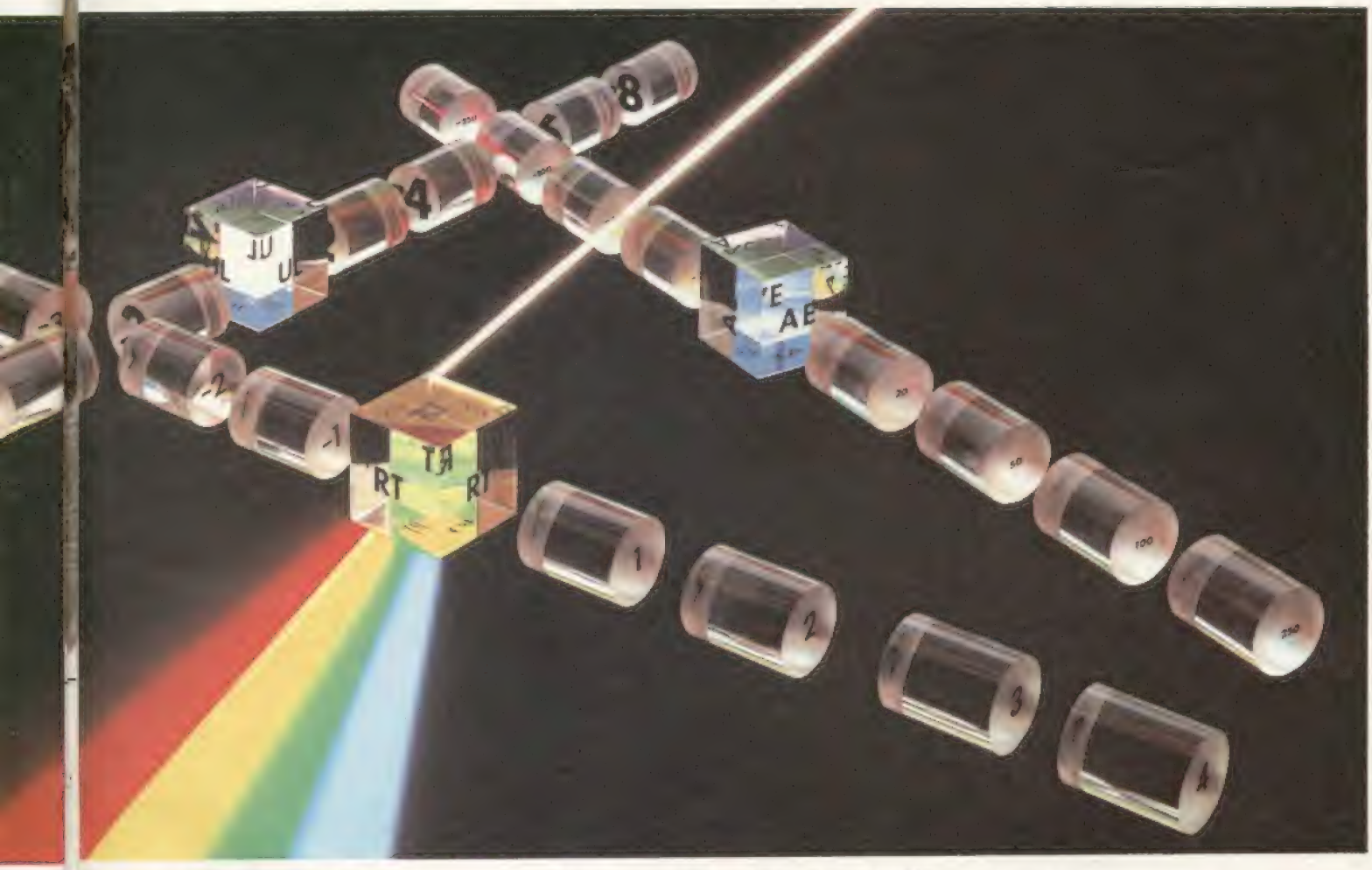
VARIÁVEIS EM AÇÃO

A maioria dos programas que você escrever utilizará diversas variáveis. Eis

um programa curto que simula a ação de uma bomba "universal" de combustível (gasolina, álcool, etc.) e que funciona de maneira muito interessante:

```
10 LET LITROS=0
20 LET CRUZ=0
30 CLS:PRINT
40 PRINT " ESCOLHA O COMBUSTIVE
L:          aLCOOL    gASOLINA
   dIESEL "
50 INPUT A$
60 IF A$="A" THEN QTY=0.3
70 IF A$="G" THEN QTY=0.2
80 IF A$="D" THEN QTY=0.4
90 CLS
100 PRINT @130,"COMBUSTIVEL  ";
A$
```

- O QUE É UMA VARIÁVEL?
- COMO SÃO UTILIZADAS AS VARIÁVEIS?
- VARIÁVEIS DE CONTROLE PARA LAÇOS **FOR...NEXT**
- O QUE É CORDÃO (STRING)
- VARIAVEIS NUMÉRICAS E ALFANUMÉRICAS
- CORDÕES VAZIOS E PROGRAMAÇÃO DE JOGOS



```
110 PRINT @199,"LITROS"
120 PRINT @256,"PRECO A PAGAR"
130 LET K=PEEK(344)
140 IF K=251 THEN CLS:GOTO 10
150 IF K=253 THEN LET CRUZ=CRUZ
+1:LET LITROS=INT((LITROS+QTY)*
10)/10
160 PRINT @206,LITROS
170 PRINT @270,CRUZ;"MIL CRUZEI
ROS"
200 GOTO 130
```

Atenção: o programa listado acima rodará apenas nos compatíveis com o TRS-Color. Para poder executá-lo também em micros da linha TRS-80, faça as seguintes modificações:

```
100 PRINT @258,"COMBUSTIVEL
    ";A$
110 PRINT @385,"LITROS"
120 PRINT @512,"PRECO A PAGAR"
160 PRINT @396,LITROS;
170 PRINT @526,CRUZ;"MIL CRU-
    ZEIROS";
```

```
10 LET litros=0
20 LET cruz=0
40 PRINT " Escolha o combusti
vel: "
45 PRINT "A(lcool)   G(asolin
a)   D(iesel)"
50 INPUT A$
60 IF A$="a" THEN  LET qty=1/
3
70 IF A$="g" THEN  LET qty=1
5
80 IF A$="d" THEN  LET qty=1/
3
 90 CLS
100 IF INKEY$="n" THEN  LET cr
uz=cruz+1: LET litros=INT ((li
tros+qty)*10)/10
105 PRINT AT 6,2;"COMBUSTIVEL
 ";A$
110  PRINT AT 9,7;"LITROS"
120 PRINT AT 9,15;litros;
130 PRINT AT 12,0;"PRECO A PAG
AR"
140 PRINT AT 12,15;cruz;" mil
cruzeiros"
150 IF INKEY$="f" THEN  CLS :
GOTO 10
200 GOTO 100
```

```
 5 CLS
10 LET LITROS=0
```

```
20 LET CRUZ=0
40 PRINT " ESCOLHA O COMBUSTIV
EL: "
45 PRINT  "A(LCOOL)   G(ASOLIN
A)   D(IESEL)"
50 INPUT A$
60 IF A$="A" THEN LET QTY=1/3
70 IF A$="G" THEN LET QTY=1/5
 80 IF A$="D" THEN LET QTY=1/3
 90 CLS
100 IF INKEY$<>"N" THEN GOTO 10
5
101 LET CRUZ=CRUZ+1
102 LET LITROS=LITROS+QTY
105 PRINT AT 6,2;"COMBUSTIVEL
";A$
110 PRINT AT 9,7;"LITROS"
120 PRINT AT 9,15;LITROS;
130 PRINT AT 12,0;"PRECO A PAGA
R"
140 PRINT AT 12,15;CRUZ;" MIL C
RUZEIROS"
150 IF INKEY$="F" THEN RUN
200 GOTO 100
```

```
5 REM ***TECLE A BARRA DE ESPAÇ
OS PARA LIGAR A BOMBA E D PARA
DESLIGA-LA***
10 LET LITROS=0
20 LET CRUZEIROS=0
30 CLS:LOCATE 10:PRINT"BOMBA DE
 COMBUSTÍVEL"
40 PRINT:PRINT:PRINT"Escolha o
seu combustível"
50 LOCATE 10,5:PRINT"1- alcool"
60 LOCATE 10,6:PRINT"2- gasolin
a"
70 LOCATE 10,7:PRINT"3- diesel"
80 LOCATE 15,12:INPUT"Sua opção
: ";C
90 IF C=1 THEN P=1/3.1
100 IF C=2 THEN P=1/4.7
110 IF C=3 THEN P=1/3.5
120 CLS
130 LOCATE 10,5:PRINT"Combustív
el ";C
140 LOCATE 10,10:PRINT"Preço a
pagar =>"
150 LOCATE 17,12:PRINT"Litros=>
"
160 IN$=INKEY$:IF IN$=CHR$(68)
THEN END
165 IF IN$<>CHR$(32)THEN 160
170 CR=CR+1000:LI=LI+P
180 LOCATE 26,10:PRINT CR
190 LOCATE 26,12:PRINT LI
200 GOTO 160
```

```
5  REM  **TECLE A BARRA DE ESPA
COS PARA RODAR A BOMBA E F PARA
 TERMINAR
10  LET LITROS = 0
20  LET CRUZEIROS = 0
30  HOME : HTAB 10: PRINT "BOMB
A DE COMBUSTIVEL"
40  PRINT : PRINT "Escolha o se
u combustivel:"
50  PRINT : HTAB 5: PRINT "1- A
LCOOL"
60  HTAB 5: PRINT "2- GASOLINA"
70  HTAB 5: PRINT "3- DIESEL"
80  VTAB 15: HTAB 10: INPUT "Op
cao =>";C
90  IF C = 1 THEN P = 1 / 3.1
100  IF C = 2 THEN P = 1 / 4.7
110  IF C = 3 THEN P = 1 / 3.5
120  HOME
130  VTAB 5: HTAB 10: PRINT "Co
mbustivel ";C
140  VTAB 10: HTAB 10: PRINT "P
reco a pagar =>"
150  PRINT : HTAB 17: PRINT "Li
tros =>"
160  GET IN$: IF IN$ =  CHR$ (7
0) THEN  END
165  IF IN$ <  >  CHR$ (32) THE
N 160
170 CR = CR + 1000:LI = LI + P
180  VTAB 10: HTAB 26: PRINT CR
190  VTAB 12: HTAB 26: PRINT  I
NT (LI * 100 + .5) / 100
200  GOTO 160
```

Vale a pena experimentar esse programa de muitas maneiras, até entender como ele funciona, pois há nele um conjunto razoável de variáveis.

A função de algumas delas é bastante evidente: a variável A, por exemplo, seleciona o tipo de combustível, através de um número (1, 2 ou 3) e determina, nas linhas 60 a 80, qual é a fração de litro do combustível escolhido que vale mil cruzados.

A tecla N aciona a bomba que mede o fluxo de combustível. O programa começa então a incrementar o número de litros de combustível (variável **CRUZ**, inicialmente zerada na linha 20, bem como a variável **LITROS**). Enquanto a tecla N estiver sendo pressionada, o programa continuará avançando; ao atingir a linha 200, ele retornará à linha 130 (100 no Spectrum), e manterá o medidor funcionando. Para terminar de abastecer, pressione a letra F.

A presença na tela de frações decimais com muitos dígitos revela a imprecisão própria do computador, quando é feita a conversão de números binários internos, para decimais comuns.

VARIÁVEIS DE CONTROLE

Assim como são utilizadas em declarações **LET**, as variáveis também podem aparecer em laços **FOR...NEXT**, como você já deve ter percebido. Neste caso, a variável que é alterada a cada repetição do laço é chamada de *variável de controle*.

Em alguns computadores, os caracteres empregados para dar nome às variáveis de controle não podem ser aplicados a variáveis numéricas (veja quadro). Nos micros da linha Sinclair, por exemplo, você pode digitar:

```
LET SCORE = 0
LET RECORDE = 0
```

para variáveis numéricas, mas não pode utilizar nomes como estes em instruções **FOR...NEXT**, tais como em:

```
FOR TEMPO = 1 TO 99999
```

Ao invés disso, você é obrigado a empregar a variável de controle com uma única letra, como por exemplo T. Os outros computadores (linhas TRS, Apple e MSX) não têm essa exigência.

VARIÁVEIS ALFANUMÉRICAS

E se, além de números, quisermos armazenar na memória do computador informações como nomes, endereços, etc.?

As variáveis capazes de armazenar um *conjunto de caracteres,* incluindo letras e números, são chamadas de variáveis alfanuméricas, em BASIC. Outra denominação usada é a de cordão (ou *string,* em inglês). Para facilitar a compreensão desse novo elemento, podemos partir de uma comparação: um cordão funciona como uma espécie de sacola de compras. Você pode enchê-la com um grande número de artigos e manejá-la como uma unidade — sem ter que ocupar-se com cada artigo separadamente. Da mesma forma, um único nome de variável alfanumérica serve para caracterizar várias memórias, simultaneamente.

Os cordões podem conter quase tudo: letras, números, sinais de pontuação, espaço em branco, e até mesmo caracteres gráficos. Eles devem estar sempre delimitados entre aspas

"NOMES E ENDEREÇOS DE CLIENTES".

"POR FAVOR ENTRE AGORA SUA RESPOSTA"

"24 de Janeiro"

"01-7346710"

"NE 135 S 181"

O exemplo seguinte também é um cordão, embora apenas alguns computadores, como o Spectrum, o TK-2000 e o MSX, permitam que se possa digitá-los desta forma diretamente no teclado: " "

A linguagem BASIC tem recursos para manipular cordões alfanuméricos de diversas maneiras. Por exemplo, existem funções e operações para juntar dois ou mais cordões uns nos outros (operação de concatenação), ou para dividi-los em dois ou mais subcordões.

Entretanto, o computador não "entende" o que está dentro de um cordão e reproduz o conteúdo deste sempre de forma exatamente igual, mesmo que ele esteja escrito em sânscrito ou alemão.

Não é possível, também, efetuar operações matemáticas com um cordão, como somas, multiplicações, divisões e outras.

Para provar isso, rode o programa abaixo:

```
10 PRINT "2+2= ";
20 FOR N=1 TO 40
25 NEXT N
30 PRINT 1+2*2
```

O conteúdo do cordão (entre aspas) é impresso como você o entrou. Na verdade, somente a linha 30 é calculada, e seu resultado, impresso (a linha 20 simula o tempo que o computador está "gastando" para pensar, antes de fornecer sua resposta). No MSX, no Apple e no TRS-80, modifique 40 para 2000, para um retardo razoável de tempo.

Mas se quiser utilizar o mesmo cordão mais de uma vez, pode economizar tempo de digitação (e espaço em memória) criando um rótulo para ele. Esse rótulo é chamado também de variável alfanumérica. Sua extensão pode variar de um computador para outro (veja quadro) mas a variável deve ser sempre seguida por um sinal de cifrão ($).

Eis aqui, por exemplo, um sistema automático de pedidos em uma lanchonete:

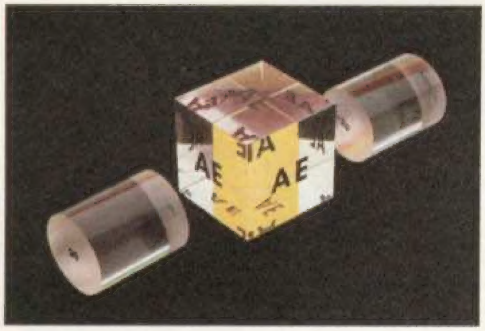
```
10 LET A$="POR FAVOR,DIGITE QU
ANTOS"
20 LET B$=" VOCE QUER"
30 PRINT A$;"HAMBURGERES";B$
35 INPUT H
40 PRINT A$;"SACOS DE FRITAS";
B$
45 INPUT S
50 PRINT A$;" MILK SHAKES";B$
55 INPUT M
60 PRINT "OBRIGADO. IRA CUSTAR
 ";(H*  15000+S*5000+M*13000);
" POR FAVOR."
```

A possibilidade de reduzir o tamanho dos programas é a razão pela qual sistemas de processamento de dados e de textos — que incluem informações e mensagens muito utilizadas, tais como "número da conta do cliente" e "preço por milhar" e "18% de ICM" — utilizam variáveis alfanuméricas.

Você também poderá empregá-las na programação de jogos. Suponhamos que, em um jogo, seja necessário a todo momento colocar na tela uma linha enorme de símbolos gráficos — o muro de uma prisão para um jogo de aventuras, por exemplo. Tudo o que você precisa fazer é digitá-lo uma única vez, "rotulá-lo" e

Variáveis: o que você pode e não pode usar

Tipo de variável	SS	TT	MX	Apple
Variável numérica	Comprimento: sem limite, todas as letras são reconhecidas.	Comprimento: máximo de 255 caracteres, mas só as duas primeiras letras são reconhecidas.	Comprimento: máximo de 255 caracteres, mas só as duas primeiras letras são reconhecidas.	Comprimento: máximo de 255 caracteres, mas só as duas primeiras letras são reconhecidas.
	No ZX-81, só letras maiúsculas. No Spectrum maiúsculas e minúsculas, mas só maiúsculas são reconhecidas.	No TRS-Color, só letras maiúsculas. No TRS-80 há conversão automática para maiúsculas.	Conversão automática para maiúsculas.	Só letras maiúsculas: não há conversão automática.
	Caracteres especiais no nome não permitidos.	Caracteres especiais no nome não permitidos.	Caracteres especiais no nome não permitidos.	Caracteres especiais no nome não permitidos.
	Espaços no nome: permitidos mas não contam	Espaços no nome: não permitidos.	Espaços no nome: não permitidos.	Espaços no nome: não permitidos.
Variável de controle p/ **FOR...NEXT**	Uma letra apenas.	Variável numérica inteira ou de precisão simples.	Variável numérica sem restrições.	Variável numérica sem restrições.
Variável alfanumérica	Uma letra apenas, seguida de $.	Como nas variáveis numéricas, seguida de $, ou declarada pela inicial em **DEFSTR**.	Como nas variáveis numéricas, seguida de $, ou declarada pela inicial em **DEFSTR**.	Como nas variáveis numéricas, seguida de $.



utilizá-lo sempre que quiser.

Movimente esse cordão por toda a tela, como mostramos no programa:

```
5 CLS
10 LET B$="                "
20 LET A$=B$+"  UMA FELIZ PASCO
A PARA VOCES"+B$
30 FOR N=1 TO 65
40 PRINT @160,MID$(A$,N,32)
50 PRINT @320,MID$(A$,66-N,32)
60 SOUND N*3,1:NEXT N
80 GOTO 30
```

Para rodar o programa acima nos micros da linha TRS-80, substitua as linhas 40, 50 e 60; assim:

```
40 PRINT @320,MID$(A$,N,32)
50 PRINT @640,MID$(A$,66-N,32)
60 NEXT N
```

```
10 LET B$="
                "
20 LET A$=B$+"     UMA FELIZ
PASCOA P  ARA VOCES  "+B$
30 FOR N=1 TO 65
40 PRINT  INK 2;AT 8,0;A$(N
TO N+31)
50 PRINT  INK 2;AT 12,0;A$(66
-N TO 97-N)
60 SOUND .02,N/2
70 NEXT N
80 GOTO 30
```

```
10 LET B$="
                "
20 LET A$=B$+"     UMA FELIZ PA
SCOA PARA VOCES  "+B$
30 FOR N=1 TO 65
40 PRINT AT 8,0;A$(N TO N+31)
50 PRINT AT 12,0;A$(66-N TO 97
-N)
70 NEXT N
80 GOTO 30
```

```
10 CLS:CLEAR 1000
20 B$=SPACE$(40)
30 A$=B$+"       UMA FELIZ PASCO
A PARA VOCES       "+B$
40 FOR N=1 TO 80
50 LOCATE 0,7:PRINT MID$(A$,N,3
9)
60 LOCATE 0,14:PRINT MID$(A$,81
-N,39)
65 FOR X=1 TO 50 : NEXT
70 NEXT
```

```
10  HOME
20 B$ = "
                    ": REM  40 ES
PACOS
30 A$ = B$ + "        UMA FELIZ
PASCOA PARA VOCES       " + B$
35  INVERSE
40  FOR N = 1 TO 80
50  VTAB 7: PRINT  MID$ (A$,N,4
0)
60  VTAB 14: PRINT  MID$ (A$,81
 - N,40)
65  FOR X = 1 TO 100: NEXT
70  NEXT
80  GOTO 40
```

Aqui você tem um cordão enorme, como mostramos na linha 20 — todos os espaços na linha 10, mais a mensagem, colocados juntos. Mais adiante, veremos como isso é feito, e como o cordão é "cortado" novamente para caber na tela. Mas, se nesse meio tempo você quiser modificar a mensagem, certifique-se de que sua nova declaração, incluindo os espaços, tem a mesma extensão da original. De outro modo você terá que descobrir o que significam os números nas linhas 40 a 55 e modificá-los.

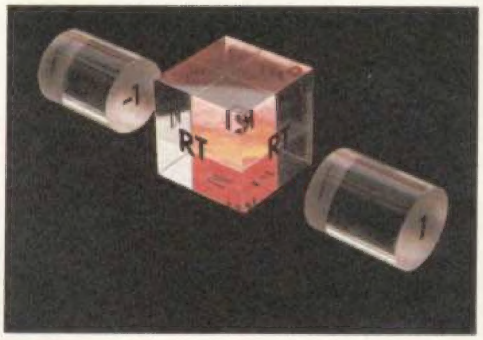
As variáveis alfanuméricas são também muito empregadas quando o programa precisa entrar uma informação que é sempre modificada. Por exemplo:

```
10 PRINT "QUAL E O SEU NOME "
20 INPUT N$
30 PRINT "ALO, ";N$
```

Neste programa a linha 10 imprime o seguinte cordão na tecla: QUAL É O SEU NOME, seguido por um ponto de interrogação. O computador espera que você entre um nome (um outro cordão) e o chama de N$. Na linha 30, ele imprime o cordão "ALO", e chama de volta o cordão N$ armazenado na linha acima. Assim será exibido na tela:

```
ALO, TOM ( ou PAULO ou PEDRO )
```

ou qualquer outro. Observe que, aqui, os cordões QUAL É O SEU NOME e ALO não são armazenados dentro de variáveis, tais como **A$** ou **QUE$** ou **ALO$,** porque eles aparecem apenas uma vez no programa, e o computador não precisa se lembrar do que seja ALO



— quando ele atinge a linha 20 simplesmente lê esse cordão e o imprime.

Ao mesmo tempo, é necessário dar um rótulo para o nome que vai ser entrado. Assim, ele poderá ser diferente cada vez que o programa for rodado. Ele é armazenado em uma variável, na linha 20 do programa, e usado na linha 30.

Existem computadores que admitem uma forma mais curta e mais clara de programação da declaração **INPUT**, através da combinação do **PRINT** "mensagem" da linha 10, com o **INPUT** da linha 20:

```
10 INPUT "QUAL E SEU NOME?" N$
20 PRINT"OLA, ";N$
```

CORDÕES VAZIOS

Outro modo de trabalhar com variáveis alfanuméricas já foi mostrado anteriormente na lição "Apontar...Fogo!" (págs. 28 e 33). Embora varie de computador para computador a forma de escrever uma variável, nesse caso, é esta:

```
20 IF A$ = "" THEN GOTO 10
```

As duas aspas sem nada entre elas são conhecidas como um cordão nulo ou vazio. A linha de programa significa: "se a entrada for igual a nada" — isto é, se nenhuma tecla estiver sendo pressionada — "volte para a linha 10 e espere até que uma tecla seja pressionada". Isto evita que o computador salte para outra parte do programa tão rapidamente que o jogador não tenha tempo de pressionar uma tecla de comando qualquer, ou de ver um resultado ou gráfico na tela.

Duas aspas com um espaço em branco entre elas, no entanto, são uma coisa muito diferente. Se você modificar a linha:

```
20 IF A$ = "  " THEN GOTO 10
```

o programa retrocederá à linha 10 apenas se o caractere entrado pelo jogador for um espaço — isto é, se ele estiver pressionando a tecla de espaço.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

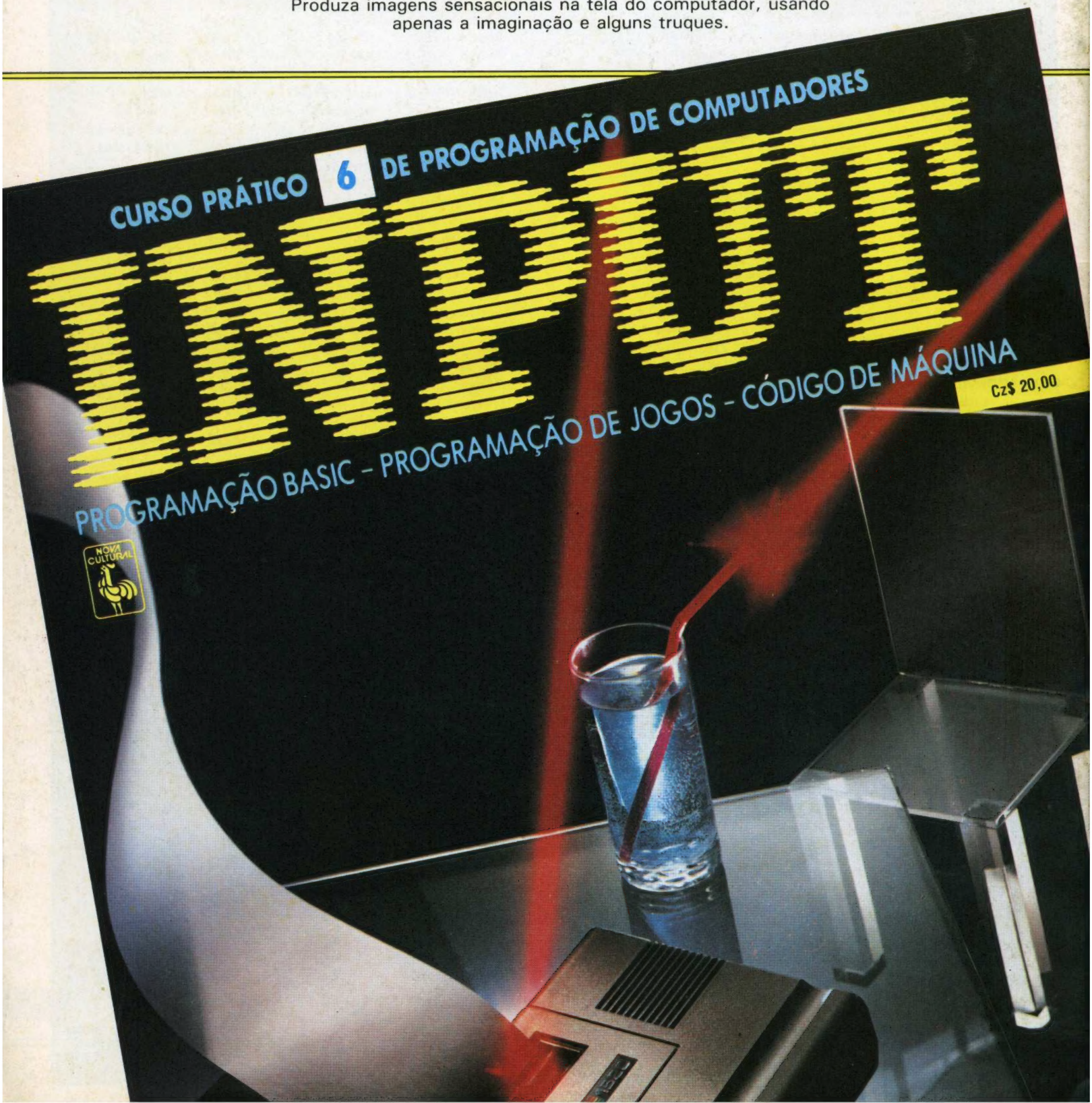
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO DE JOGOS
Participe da "guerra nas estrelas"... no computador. Desenhe na tela os elementos do jogo e construa a sua blindagem.

LINGUAGEM DE MÁQUINA
Conheça o coração de um micro. O que é e o que faz uma UCP. Como o micro acompanha o desenvolvimento de um programa.

PROGRAMAÇÃO BASIC
Produza imagens sensacionais na tela do computador, usando apenas a imaginação e alguns truques.